Evaluation-Function-based Model-free Adaptive Fuzzy Control

Agus Naba

Study Program of Instrumentation,
Department of Physics, Faculty of Mathematics and Natural Sciences,
University of Brawijaya, Jalan Veteran, Malang 65145, Indonesia
E-mail: anaba@ub.ac.id

Abstract. Designs of adaptive fuzzy controllers (AFC) are commonly based on the Lyapunov approach, which requires a known model of the controlled plant. They need to consider a Lyapunov function candidate as an evaluation function to be minimized. In this study these drawbacks were handled by designing a model-free adaptive fuzzy controller (MFAFC) using an approximate evaluation function defined in terms of the current state, the next state, and the control action. MFAFC considers the approximate evaluation function as an evaluative control performance measure similar to the state-action value function in reinforcement learning. The simulation results of applying MFAFC to the inverted pendulum benchmark verified the proposed scheme's efficacy.

Keywords: *adaptive fuzzy control; evaluation function; Lyapunov approach; model-free adaptive control; reinforcement learning.*

1 Introduction

Many designs of adaptive fuzzy controllers (AFC) are based on the Lyapunov approach, assuming the existence of a Lyapunov function for the control problem to be solved. The Lyapunov function getting smaller is seen as an indication of better control performance. The most important stage in the Lyapunov approach is to ensure that the first derivative of the Lyapunov function candidate (LFC) is either negative-definite or semi-definite [1-4]. This stage requires a known plant model or, at least, a known plant model structure.

The regular approach in model-based design of AFCs is that the engineer a priori defines the LFC simply in terms of the distance between the current state and the goal state. A smaller error is always assumed to indicate better control performance. However, in many control problems a smaller error does not indicate better control performance, i.e. it is not always evaluative. Thus, action selection based on smaller errors can misdirect and lead to non-goal states. Generally speaking, errors are more instructional than evaluative [5]. For this reason, they are not universally suitable as an evaluative measure of control performance. This is a flaw in model-based design of AFCs.

Received April 6th, 2016, Revised June 23rd, 2016, Accepted for publication December 16th, 2016.
 ISSN: 2337-5779, DOI: 10.5614/j.eng.technol.sci.2016.48.6.4

The reinforcement learning (RL) approach can cope with the drawbacks of model-based design of AFCs. It is model-free, generally applicable, and uses an evaluative measure for its control performance. In RL, evaluative information about control performance cannot be provided a priori by the engineer but only by the environment in terms of evaluative future rewards [6]. The environment is identical to the controlled plant. RL uses the received rewards to learn a value function on-line that represents a goodness measure of states due to the execution of the control action sequence. Applied to a control problem, the objective of RL is to improve the control performance as measured by the value function, by generating appropriate actions. The main drawback of RL is time-consumingness in learning the precise value function through lots of trial-and-error interactions with the plant.

This paper presents an evaluation-function-based model-free adaptive fuzzy control scheme with which the fuzzy controller can achieve faster convergence. Many real control problems comprise simpler sub-problems. In the case of a relatively simple control problem, an approximate evaluative goodness measure of the control action executed at the current state can be easily represented in terms of the distance between the next state and the goal state and the executed action. As the states get closer to the goal state, the action must be close to zero. Such an approximate evaluative measure is used as the evaluation function in the proposed adaptive fuzzy control scheme.

Some relevant adaptive methods have been proposed in [1,5,7-14]. However, they assume a known model or at least a known structure of the plant model. They are not applicable to plants that cannot be represented by the assumed model structure. Previous research on the same issue as addressed in this paper is reported in [15-18]. Although the proposed systems work, their performance is poor. The goal in this study was to achieve better performance than these previous attempts. The Lyapunov approach is introduced during the adjustment of the fuzzy controller based on an approximate evaluation function.

This paper is organized as follows: Section 2 briefly discusses RL. Section 3 presents the proposed model-free adaptive fuzzy control scheme. Section 4 discusses the results of applying the proposed scheme to solve a benchmark control problem. The last one, Section 5, concludes the paper.

2 Reinforcement-Learning Adaptive Control

Figure 1 shows the adaptive control problem from the point of view of RL. The controller and the plant correspond to an agent and an environment respectively. The use of a reward as a goodness measure of action makes RL very different

from the adaptive control approach. The reward is a weak goodness measure, which can simply be a *success* or a *failure* feedback signal given by the plant.

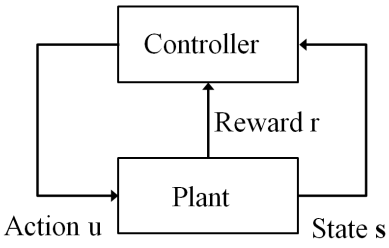

Figure 1 Reinforcement-learning model in control problem.

In Figure 1, the controller receives a reward $r(k+1)$ after executing an action $u(k)$ at a state $\mathbf{s}(k)$, where k is the discrete-time index. Its goal is to maximize the value function

$$V(\mathbf{s}(k)) = max_{\bar{u}(k)} \sum_{i=0}^{\infty} \gamma^i \, r(k+1+i) \tag{1}$$

where γ is a discount factor against a future reward and $\bar{u}(k)$ is a policy to be optimized, defined as a sequence of actions:

$$\bar{u}(k) = (u(k), u(k+1), u(k+2), \dots).$$

The value function measures the long-term desirability of states after execution of a policy, in contrast with the reward, which measures the immediate, intrinsic desirability of plant states. The controller must choose actions leading to the states of highest values of $V(\mathbf{s}(k))$, rather than highest rewards $r(k)$. The actions with the highest values will result in the greatest $V(\mathbf{s}(k))$ in the long run, leading to the optimal policy.

The optimal policy can be obtained using either a value-function approach (VFA) or a policy-search approach (PSA).

3 Value Function Approach

VFA considers Eq. (1) as an optimization problem according to the Bellman equation, can be formulated as in Eq. (2),

$$V(\mathbf{s}(k)) = max_{\bar{u}(k)} \left[r(k+1) + \gamma V\left(h(\mathbf{s}(k), u(k)) \right) \right] \tag{2}$$

where $h(\mathbf{s}(k), u(k)) = \mathbf{s}(k+1)$ is a function of plant dynamics. The optimal policy can be formulated as:

$$u^*(\mathbf{s}(k)) = \arg max_{\bar{u}(k)} \left[r(k+1) + \gamma V \left(h(\mathbf{s}(k), u(k)) \right) \right]. \quad (3)$$

Eq. (3) is not applicable, as $h(\mathbf{s}(k), u(k))$ is unknown. Alternatively, one can use a state-action-value function (a.k.a. Q-function) as shown in Eq. (4),

$$Q(\boldsymbol{s}(k), u(k)) = r(k+1) + \gamma Q(\mathbf{s}(k), u^*(s(k+1))) \quad (4)$$

Where in Eq. (5), $u^*(s(k+1))$ is a model-free optimal policy deduced by:

$$u^*(\mathbf{s}(k)) = \arg max_{\bar{u}(k)} Q(\mathbf{s}(k+1), u(k+1)). \quad (5)$$

Of course, the Q-function initially provides poor policies. It can be updated using the temporal difference (TD) method [19]. Exploration of appropriate actions in the early stages is necessary, as the current policy still lacks optimal actions. Choosing random actions within a small fraction of time provides more chance to find better actions. Figure 2 illustrates how VFA adjusts the controller.

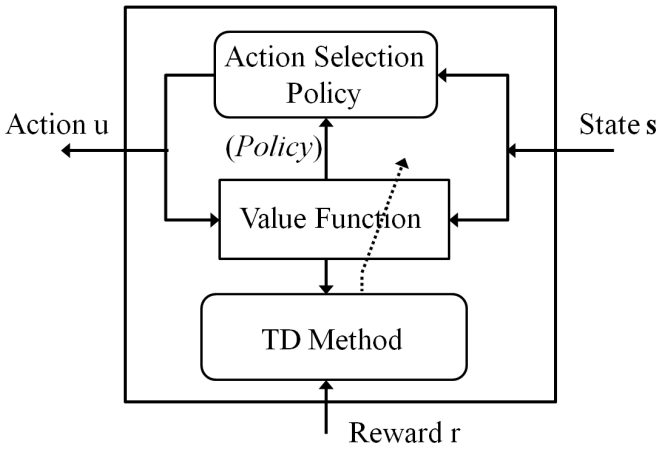

Figure 2 Adaptive controller with value function approach.

In Eq. (6), the TD method updates the Q-function by:

$$Q(\mathbf{s}(k), u(k)) \leftarrow Q(\mathbf{s}(k), u(k)) + \alpha\delta(k) \quad (6)$$

where α is the learning rate and $\delta(k)$ is the TD error:

$$\delta(k) = r(k) + \gamma Q(\mathbf{s}(k+1), u^*(\mathbf{s}(k+1))) - Q(\mathbf{s}(k), u(k)).$$

In case of discrete states and actions, the Q-function can simply be represented by a look-up table that maps a state-action pair to its value. If states and actions are continuous, a neural network or a fuzzy system can be used as the function

approximator for the Q-function whose weights, instead of Q-values, can be updated using the TD method.

Let $\mathbf{\Omega} = \{\omega\}$ be a weight vector of the Q-function $Q_{\mathbf{\Omega}}$. From Eqs. (7) and (8), the TD method updates each element of $\mathbf{\Omega}$ by:

$$\omega(t) \leftarrow \omega(t) + \eta\delta(t)G(t) \tag{7}$$

$$G(t) \leftarrow \lambda\gamma G(t) + \frac{\partial Q_{\mathbf{\Omega}}(t)}{\partial \omega(t)} \tag{8}$$

where η is the learning rate, $0 \leq \lambda \leq 1$, and t is the continuous-time index. This TD update rule requires the initial value $G(0)$.

4 Policy Search Approach

In the policy search approach (PSA), as shown in Figure 3, the policy can be represented by any function approximator, such as a neural network or a fuzzy system, taking the current state $\mathbf{s}$ as the input and the action u as the output. The reward r is received after execution of action u at current state $\mathbf{s}$. The estimation of the value function is based on the information of current state $\mathbf{s}$, action u, and reward r at the next time step. The estimated value function is not directly used for action selection, but instead as an evaluation function for tuning the parameters of the policy. It can have the form of either $V(\mathbf{s})$ or $Q(\mathbf{s}, u)$.

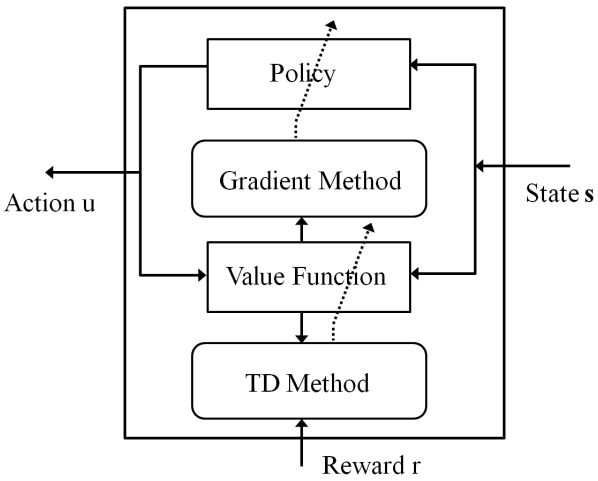

Figure 3 Adaptive controller using policy search approach.

Let the policy be represented by $u(t) = u_{\mathbf{w}}(t)$, where $\mathbf{w} = \{w\}$ is a weight vector and $\partial u(t)/\partial w(t)$ exists. In Eq. (9), using $V(\mathbf{s}(t))$ as the evaluation function to be maximized, one can update w by:

$$w(t) \leftarrow w(t) + \eta \frac{\partial V(\mathbf{s}(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w(t)}. \tag{9}$$

Unfortunately, the partial derivative $\partial V(\mathbf{s}(t))/\partial u(t)$ does not exist. In [20,21], $V(\mathbf{s}(t))$ is assumed to be indirectly dependent on $u(t)$, and its partial derivative with respect to $u(t)$ is approximated as following Eq. (10) :

$$\frac{\partial V(\mathbf{s}(t))}{\partial u(t)} \approx \frac{\Delta V(\mathbf{s}(t))}{\Delta u(t)} = \frac{V(\mathbf{s}(t)) - V(\mathbf{s}(t-\Delta t))}{u(t) - u(t-\Delta t)}. \tag{10}$$

The update rule in Eq. (9) seems ineffective in tuning weight vector $\mathbf{w}$. The accuracy of evaluation function $V(\mathbf{s}(t))$ is not guaranteed during the learning process, whereas the computation of the partial derivative $V(\mathbf{s}(t))$ with respect to $u(t)$ requires $V(\mathbf{s}(t))$ to be accurate. On the other hand, $V(\mathbf{s}(t))$ only provides the value of state $\mathbf{s}(t)$, independent of the execution of action $u(t)$. Hence, it is not acceptable to assume that $V(\mathbf{s}(t))$ is always dependent on $u(t)$, either directly or indirectly. Consequently, there can be situations where the approximation $\Delta V(\mathbf{s}(t))/\Delta u(t)$ in (10) is meaningless.

This problem can be solved by using the state-action value function $Q(\mathbf{s}(t), u(t,))$ instead of the value function $V(\mathbf{s}(t),)$ as the evaluation function. Let $\mathbf{w} = \{w\}$ now be a parameter vector of performance measure ρ, represented as a function of the Q-function (i.e. $\rho(Q)$). In Eq. (11), using $\rho(Q))$ as the evaluation function to be maximized, the policy search approach can update w by the following gradient ascent method [22] :

$$w(t) \leftarrow w(t) + \eta \frac{\partial \rho(Q(\mathbf{s}(t), u(t)))}{\partial w(t)}. \tag{11}$$

Assuming that $\partial \rho(Q)/\partial u$ and $\partial u/\partial w$ exist, the policy search approach can also update w following Eq. (12),

$$w(t) \leftarrow w(t) + \eta \frac{\partial \rho(Q(\mathbf{s}(t), u(t)))}{\partial u(t)} \frac{\partial u(t)}{\partial w(t)}. \tag{12}$$

5 Limitations of Reinforcement Learning

Various control problems have been successfully solved using VFA and PSA [21-23]. However, their performance suffers for the following reasons:

1. The precise value function needs to be found through lots of trial-and-error interactions between the controller and the plant.

2. Decision on the best action to take can be very sensitive to small arbitrary changes in the estimated value function.
3. The value function is only associated with a fixed goal state. This leads to the limitation that the optimal policy derived from the value function is only the best when applied to achieve the fixed goal state.

6 Proposed Adaptive Fuzzy Control

The proposed fuzzy adaptive control system is described in Figure 4. It involves manipulating a fuzzy controller to improve its performance as measured by an evaluation function. Generally speaking, the controller architectures in Figures 3 and 4 are similar in that both use the evaluation function and the gradient method in tuning the controller parameters. The update rule used in Figure 4 is similar to Eq. (12). The main idea in this study was to replace the performance measure $\rho(Q)$ by an approximate value function considered as the evaluation function to be minimized. The evaluation function is readily available.

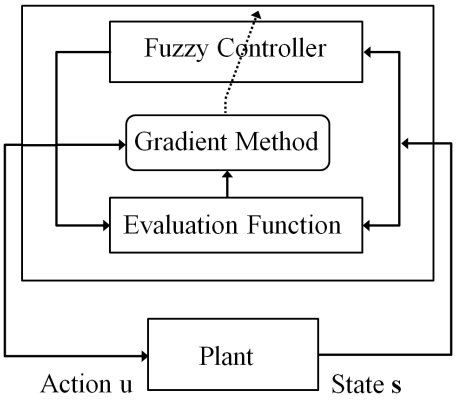

Figure 4 Model-free adaptive fuzzy control system based on evaluation function.

7 Fuzzy Controller

In this study, the fuzzy controller was developed using a zero-order Takagi-Sugeno fuzzy system, mapping an input vector, $\mathbf{s} = [s_1, s_2, \ldots, s_m]^T$, to a scalar output, f. Let us define M_i fuzzy sets $F_i^j, j = 1, \ldots, M_i$, for each input s_i, then the fuzzy controller can be written by a set of if-then rules following Eq. (13),

$$\begin{aligned} &\mathrm{R}^{\mathrm{r}}: \text{IF } s_1 \text{ is } H_i^r \text{ and } \ldots \text{ and } s_m \text{ is } H_m^r \\ &\text{THEN } f \text{ is } f^r, (r = 1, \ldots, N) \end{aligned} \tag{13}$$

where $H_i^k = \{F_i^1, \ldots, F_i^{M_i}\}, i = 1, \ldots, n$, f^r is the crisp output of the r-th rule, and N is the total number of rules.

Using the product inference engine and the singleton fuzzifier [1], the fuzzy controller is given by Eq. (14).

$$f(\mathbf{s}) = \frac{\sum_{r=1}^{N} f^r \mu_r(\mathbf{s})}{\sum_{r=1}^{N} \mu_r(\mathbf{s})} \tag{14}$$

where

$$\mu_r(\mathbf{s}) = \prod_{i=1}^{m} \mu_{H_i^k}(s_i), \mu_{H_i^k} \in \left\{\mu_{F_i^1}, \ldots, \mu_{F_i^{M_i}}\right\}$$

where $\mu_{F_i^l}$ denotes the membership function of fuzzy set F_i^l. In a more compact form it can be rewritten as follows:

$$f(\mathbf{s}) = \mathbf{z}^T \mathbf{b}(\mathbf{s}) \tag{15}$$

where $\boldsymbol{z} = [f^1, \ldots, f^N]^T$ denotes a parameter vector and $b(\mathbf{s}) = [b_1(\mathbf{s}), \ldots, b_N(\mathbf{s})]^T$ is a set of fuzzy basis functions whose r-th element is defined by:

$$b_r(\mathbf{s}) = \frac{\mu_r(\mathbf{s})}{\sum_{j=1}^{N} \mu_j(\mathbf{s})},$$

assuming $\sum_{j=1}^{N} \mu_j(\mathbf{s}) \neq 0$ for all $\mathbf{s}$.

The fuzzy controller in Eq. (15) is used to represent control action $u(\mathbf{s}(t)) = f_{\mathbf{z}}(\mathbf{s}(t))$ in Figure 4.

For convenience, in the following explanation the argument $\mathbf{s}(t)$ in any function is dropped and replaced simply with t, when necessary.

8 Evaluation Function

Typical adaptive control designs define the measure of control performance a priori [1,7,13,14,24,25]. Plant output errors (i.e. the difference between the current state and the goal state) getting smaller is seen as a direct indication that the executed sequence of actions is 'correct' in that it is on the right path leading to the goal state eventually. Otherwise, the executed actions are blamed and alternative actions are to be explored. Hence, the plant output error is used as a control performance measure.

Generally speaking, according to RL's viewpoint the evaluative information feedback that will be received after executing an action should depend on at least: the action itself, the state at which the action is executed, and the result of

executing the action (i.e. the next state). Obviously, the plant output error does not meet such criteria and therefore cannot be considered as universally evaluative. In other words, smaller errors do not necessarily indicate better control performance. Some methods for solving control problems based on the RL approach have been proposed [18,20,22,23]. They need real-time learning of the value function.

In this study, the value function is approximated by a function that meets the above criteria, as shown in Eq. (16).

$$I\big(\mathbf{s}(t),u(t)\big)=\frac{1}{2}(\mathbf{s}(t+\Delta t)^T(\mathbf{s}(t+\Delta t)+\alpha u^2(t)). \tag{16}$$

$u(t)$ is the action executed at state $\mathbf{s}(t)$. Δt is the elapsed time between time steps. α is the weighting factor for action $u(t)$. The goal state is assumed at $\mathbf{s_{goal}}(t)=\mathbf{0}$. In case of a non-zero or changing goal state $\mathbf{s}(t)$ is replaced by error vector $\mathbf{e}(t)=\mathbf{s}(t)-\mathbf{s_{goal}}(t)$.

We suppose that $I(\mathbf{s},u)$ has a role similar to that of state-action-value function $Q(\mathbf{s},u)$ in (4). Given $\mathbf{s}(t)$, the fuzzy controller produces and executes action $u(t)$ and drives the plant to the next state $\mathbf{s}(t+\Delta t)$. The performance of the fuzzy controller at state $\mathbf{s}(t)$ is represented by $I\big(\mathbf{s}(t),u(t)\big)$.

The proposed adaptive fuzzy control in Figure 4 uses $I\big(\mathbf{s}(t),u(t)\big)$ as the evaluation function at the next time step $t+\Delta t$ to evaluate action $u(t)$ executed at $\mathbf{s}(t)$ and then updates the parameters of fuzzy controller $\mathbf{z}(t)$. We assume that a smaller $I(\mathbf{s},u)$ implies better control performance. In the steady states near the goal state, actions $u(t)$ must be close to zero.

Although the next state $\mathbf{s}(t+\Delta t)$ is unknown at time t, it does not mean that the adaptive fuzzy control in Figure 4 will not work. In reality, to follow the aforementioned scenario, the adaptive fuzzy control may consider the previous time $t-\Delta t$ as the 'current' time step and the current time t as if the 'next' time step. Given both the 'current' action and the 'next' state, $I\big(\mathbf{s}(t-\Delta t),u(t-\Delta t)\big)$ can be computed and considered as the performance measure for the 'current' action $u(t-\Delta t)$. After $I\big(\mathbf{s}(t-\Delta t),u(t-\Delta t)\big)$ is available, it can be used to update the fuzzy controller parameters $\mathbf{z}(t-\Delta t)$. Using the updated parameters, the fuzzy controller generates the 'next' action $u(t)$ at the 'next' state $\mathbf{s}(t)$. In this way, the proposed adaptive fuzzy control can be implemented in real applications.

9 Gradient-based Update Rule

The proposed adaptive fuzzy control learns to produce appropriate actions by adjusting $\mathbf{z}(t)$ using evaluation function $I\big(\mathbf{s}(t),u(t)\big)$. This adjustment takes effect on both the action $u(t)=f_{\mathbf{z}(t)}(\mathbf{s}(t))$ and the evaluation function itself. For the sake of clarity, we use the notation $I_{\mathbf{z}(t)}$, in place of $I\big(\mathbf{s}(t),u(t)\big)$, to represent the goodness of $\mathbf{z}(t)$ at state $\mathbf{s}(t)$ and then rewrite Eq. (16) following Eq. (17):

$$I_{\mathbf{z}(t)}=\frac{1}{2}E^2(t) \tag{17}$$

where $E(t)=\sqrt{(\mathbf{s}(t+\Delta t)^T(\mathbf{s}(t+\Delta t)+\alpha u^2(t))}$.

The fuzzy adaptive control in Figure 4 adjusts the r-th element of $\mathbf{z}(t)$ by using the gradient descent method:

$$\frac{dz_r(t)}{dt}=-\eta\frac{\partial I_{\mathbf{z}(t)}}{\partial z_r(t)} \tag{18}$$

where η is a positive-definite step size. Assuming that the partial derivative of $I_{\mathbf{z}(t)}$ with respect to $z_r(t)$ exists, the update rule in Eq. (18) is a natural strategy that allows $\mathbf{z}(t)$ to converge to a local optimal point of the evaluation function $I_{\mathbf{z}(t)}$. Unfortunately, $\partial I_z/\partial z_r$ is unknown. To solve this problem, we apply the following chain rule:

$$\frac{\partial I_{\mathbf{z}(t)}}{\partial z_r(t)}=E(t)\frac{\partial E(t)}{\partial u(t)}\frac{\partial u(t)}{\partial z_r(t)}. \tag{19}$$

The partial derivative $\partial E/\partial u$ is difficult to compute, even if the plant dynamics are known. In this study it is approximated by

$$\frac{\partial E(t)}{\partial u(t)}\approx\frac{\Delta E(t)}{\Delta u(t)}=\frac{E(t)-E(t-\Delta t)}{u(t)-u(t-\Delta t)}.$$

Substituting this approximation into Eq. (19) and then Eq. (19) into Eq. (18), we obtain the following update rule:

$$\frac{\partial z_r(t)}{\partial t}\approx-\eta E(t)\frac{\Delta E(t)}{\Delta u(t)}\frac{\partial u(t)}{\partial z_r(t)}. \tag{20}$$

Given fuzzy controller in Eq. (15), the partial derivative of $u(t)$ with respect to $z_r(t)$ is determined by

$$\frac{\partial u(t)}{\partial z_r(t)}=\frac{\partial f_{z(t)}}{\partial z_r(t)}=b_r(t).$$

In this study, the method of tuning $\mathbf{z}(t)$ based on update rule Eq. (20) is referred to as the approximated gradient descent method (AGDM).

10 Update Rule with Failure Detector

The update rule in Eq. (20) is a crude approximation, ignoring and missing any state change between consecutive time steps. In addition, it is undefined when $\Delta u(t)$ is very small or zero. In [13-16], we have proposed a heuristic approach to cope with the drawbacks of Eq. (20) as follows:

$$\Delta \mathbf{z}(t) = -c(t)\eta E(t)\text{sign}\left(\frac{\Delta E(t)}{\Delta u(t)}\right)\mathbf{b}(t)\Delta t \tag{21}$$

Where $c(t)$ is referred to as a failure detector:

$$c(t) = \begin{cases} 1, & \text{if } I_{\mathbf{z}(t)} > I_{\mathbf{z}(t-\Delta t)} \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

The use of $\text{sign}\left(\frac{\Delta E(t)}{\Delta u(t)}\right)$ in Eq. (21) is intended to avoid division by a small or zero value of $\Delta u(t)$.

The role of $c(t)$ can be intuitively explained as follows. When the value $I_{\mathbf{z}(t)}$ decreases, then the current $\mathbf{z}(t)$ is considered 'good'. In such a 'good' state it is reasonable to keep $\mathbf{z}(t)$ unchanged by setting $c(t) = 0$ since any update of $\mathbf{z}(t)$ may make the situation worse. Otherwise, $c(t) = 1$.

11 Proposed Update Rule with Lyapunov Approach

Update rule Eq. (21) with failure detector Eq. (22) has been applied in our previous researches [13-16]. Its performance suffers since the failure detector $c(t)$ is defined using only two possible values, i.e. 0 or 1. According to the gradient method, $c(t)$ can have any positive value. Empirically, however, this often does not work.

To overcome this drawback, in this paper the following update rule is proposed:

$$\Delta \mathbf{z}(t) = c_L(t)\eta E(t)\text{sign}\left(\frac{\Delta E(t)}{\Delta u(t)}\right)\mathbf{b}(t)\Delta t \tag{23}$$

where $c_L(t)$ is chosen according to the Lyapunov criteria in that it must lead to a negative derivative of the evaluation function. In order to achieve this condition, we propose the following scenario of chosing $c_L(t)$ as shown in Eq. (24).

$$c_L(t) = \begin{cases} -c_1, & \text{if } \left(\Delta I_{\mathbf{z}(t)} \geq 0 \text{ and } I_1(t) \geq 0 \text{ and } I_2 \geq 0\right) \\ & \text{or } \left(\Delta I_{\mathbf{z}(t)} \geq 0 \text{ and } I_1(t) < 0 \textit{ and } I_2 \geq 0\right) \\ c_2, & \left(\Delta I_{\mathbf{z}(t)} \geq 0 \text{ and } I_1(t) \geq 0 \text{ and } I_2 < 0\right) \\ 0, & \text{otherwise,} \end{cases} \tag{24}$$

where

$$\begin{aligned} \Delta I_{\mathbf{z}(t)} = &\quad I_{\mathbf{z}(t)} - I_{\mathbf{z}(t-\Delta t)} \\ I_2(t) = &\quad \alpha u(t)\eta E(t)\text{sign}\left(\frac{\Delta E(t)}{\Delta u(t)}\right)\mathbf{b}^T(t)\mathbf{b}(t)\Delta t \\ I_1(t) = &\quad \Delta I_{\mathbf{z}(t)} - I_2(t), \end{aligned} \tag{25}$$

and $c_1 > c_2 > 0$.

$c_L(t)$ can take any value rather than only 1 or 0. The explanation of the role of $c_L(t)$ is almost the same as that of $c_L(t)$ in Eq. (22). $\mathbf{z}(t)$ is updated at the three possible conditions shown in Eq. (24); otherwise, it is left unchanged by setting $c_L(t) = 0$.

The adaptive control using update rule Eq. (23) can be implemented with the following steps:

1. initialize $\mathbf{s}$, $\mathbf{z}$, and $I_{\mathbf{z}}$
2. get the current state, $\mathbf{s}(t)$
3. using the current $\mathbf{z}(t)$, apply control action $u(t)$ to the plant
4. get the next state, $\mathbf{s}(t + \Delta t)$
5. compute the evaluation function $I_{\mathbf{z}(t)}$ and $\Delta I_{\mathbf{z}(t)}$ using Eq. (25)
6. set $c_L(t) = 1$ and compute $\Delta\mathbf{z}(t)$ using Eq. (23)
7. compute $I_2(t)$ using Eq. (25)
8. compute $I_1(t)$ using Eq. (25)
9. using $\Delta I_{\mathbf{z}(t)}$, $I_2(t)$, and $I_1(t)$, check the condition for $c_L(t)$ according to Eq. (24)
10. choose the best $c_L(t)$ according to Eq. (24)
11. compute $\Delta\mathbf{z}(t)$ again using the best $c_L(t)$
12. update the controller paremeters by $\mathbf{z}(t + \Delta\text{t}) = \mathbf{z}(t) + \Delta\mathbf{z}(t)$
13. set $t = t + \Delta\text{t}$
14. go to the step 2.

12 Convergence Analysis

The evaluation function $I_{\mathbf{z}}$ in Eq. (17) is positive definite. In this paper $I_{\mathbf{z}}$ is considered a Lyapunov function candidate and its first time derivative, $\Delta I_{\mathbf{z}(t)}$, is to be made negative at all times to achieve

$$I_{\mathbf{z}(t)} \to 0,$$

which leads to the desired condition $\mathbf{s}(t) \to 0$.

Taking the first time derivative of $I_{\mathbf{z}}$, we obtain

$$\Delta I_{\mathbf{z}(t)} = \mathbf{s}^T(t+t)\Delta\mathbf{s}(t+\Delta t) + \alpha u(t)\Delta u(t). \tag{26}$$

Whereas the term $\mathbf{s}^T(t+\Delta t)\Delta\mathbf{s}(t+\Delta t)$ includes unknown plant dynamics, the last term $\Delta u(t)$ in Eq. (26) is of great importance as it can be manipulated to make $\Delta I_{\mathbf{z}(t)}$ negative, either definite or semi-definite, as explained below.

Given the fuzzy controller in Eq. (15), i.e. $u(t) = \mathbf{z}^T(t)\mathbf{b}(t)$, we obtain

$$\Delta u(t) = \mathbf{z}^T(t)\Delta\mathbf{b}(t) + \Delta\mathbf{z}^T(t)\mathbf{b}(t). \tag{27}$$

Using Eq. (27), Eq. (26) can be rewritten as in Eq. (28).

$$\Delta I_{\mathbf{z}(t)} = I_1(t) + I_2(t) \tag{28}$$

where

$$\begin{aligned} I_1(t) &= \mathbf{s}^T(t+\Delta t)\Delta\mathbf{s}(t+\Delta t) + \alpha u(t)\mathbf{z}^T(t)\Delta\mathbf{b}(t) \\ I_2(t) &= \alpha u(t)\Delta\mathbf{z}^T(t)\mathbf{b}(t). \end{aligned} \tag{29}$$

Although $I_1(t)$ is unknown, $\Delta I_{\mathbf{z}(t)}$ can be estimated by

$$\Delta I_{\mathbf{z}(t)} \approx I_{\mathbf{z}(t)} - I_{\mathbf{z}(t-\Delta t)}$$

which can then be used to estimate $I_1(t)$ by

$$I_1(t) \approx \Delta I_{\mathbf{z}(t)} - I_2(t). \tag{30}$$

Note that $I_2(t)$ is computable and at our disposal. Modifying $I_2(t)$ is the only strategy available to achieve negative values of $\Delta I_{\mathbf{z}(t)}$. This can be done by appropriately choosing $\Delta\mathbf{z}^T(t)$.

Each of $\Delta I_{\mathbf{z}(t)}$, $I_2(t)$, and $I_1(t)$ will be of one of the two possible values, i.e. ≥ 0 or < 0, resulting in eight candidate conditions. Among those conditions, two conditions will never occur. $\Delta I_{\mathbf{z}(t)}$ is impossible to be negative when both $I_1(t)$ and $I_2(t)$ are greater than or equal to zero. On the contrary, $\Delta I_{\mathbf{z}(t)}$ is impossible to be positive or zero when both $I_1(t)$ and $I_2(t)$ are negative. The six remaining conditions that can take place are as follows:

1. $\Delta I_{\mathbf{z}(t)} < 0$ and $I_1(t) < 0$ and $I_2(t) < 0$
2. $\Delta I_{\mathbf{z}(t)} < 0$ and $I_1(t) \geq 0$ and $I_2(t) < 0$
3. $\Delta I_{\mathbf{z}(t)} < 0$ and $I_1(t) < 0$ and $I_2(t) \geq 0$
4. $\Delta I_{\mathbf{z}(t)} \geq 0$ and $I_1(t) < 0$ and $I_2(t) \geq 0$
5. $\Delta I_{\mathbf{z}(t)} \geq 0$ and $I_1(t) \geq 0$ and $I_2(t) \geq 0$
6. $\Delta I_{\mathbf{z}(t)} \geq 0$ and $I_1(t) \geq 0$ and $I_2(t) < 0$.

When condition 1, 2 or 3 takes place, the Lyapunov criteria is met. Each of these conditions can be an indication that the vector $\mathbf{z}(t)$ is pointing in the 'right' direction, by which the goal state will be achieved. Therefore, at these conditions it is reasonable to keep the controller parameters $\mathbf{z}(t)$ unchanged by setting $c_L(t) = 0$ in Eq. (23). The above strategy makes $I_2(t) = 0$ and may cause $\Delta I_{\mathbf{z}(t)}$ being positive, which makes another condition occur. To solve this problem, different strategies are required, as explained below.

At condition 4, 5 or 6, the Lyapunov criteria is not met due to the undesired condition $\Delta I_{\mathbf{z}(t)} \geq 0$. The update rule in Eq. (23) works to make $\Delta I_{\mathbf{z}(t)}$ negative whenever one of the conditions 4, 5, and 6 occurs. This is done by assigning with as different values to $c_L(t)$ (see Eq. (24)), depending on the condition occurring.

Before deciding the best $c_L(t)$ to choose, the condition must be known, which requires the information of $I_2(t)$, $\Delta I_{\mathbf{z}(t)}$, and $I_1(t)$. According to Eq. (30), $I_2(t)$ must be computed firstly, then $\Delta I_{\mathbf{z}(t)}$, and finally, $I_1(t)$.

In order to compute $I_2(t)$, initially set $c_L(t)$ to 1 and then compute $\Delta\mathbf{z}(t)$ using Eq. (23). After introducing $\Delta\mathbf{z}(t)$ into Eq. (29), we obtain Eq. (31).

$$I_2(t) = \alpha u(t)\eta E(t)\text{sign}\left(\frac{\Delta E(t)}{\Delta u(t)}\right)\mathbf{b}^T(t)\mathbf{b}(t)\Delta t \tag{31}$$

which is the same as $I_2(t)$ given in Eq. (25).

When condition 4 occurs, $\Delta\mathbf{z}(t)$ points in the wrong direction, indicated by $I_2(t) \geq 0$, which is the only contributor to condition $\Delta I_{\mathbf{z}(t)} \geq 0$. Obviously, in such a condition, setting $c_L(t) = -c_1$, where $c_1 > 0$, is the best strategy, which reverses the direction of the vector $\Delta\mathbf{z}(t)$ to the 'right' direction, resulting in $I_2(t) < 0$.

When condition 5 occurs, the chosen strategy is the same as that at condition 4, but for a somewhat different reason. Depending on how large $I_1(t)$ is compared to $I_2(t)$, condition $I_2(t) < 0$ due to the adjustment of $\mathbf{z}(t)$ may not take effect at condition $\Delta I_{\mathbf{z}(t)} \geq 0$. However, keeping condition $I_2(t) < 0$ is the only known strategy that can be done to achieve $\Delta I_{\mathbf{z}(t)} < 0$.

At condition 6, $\Delta I_{\mathbf{z}(t)}$ is positive, but $I_2(t)$ is negative already, indicating that $\Delta\mathbf{z}(t)$ is pointing in the 'right' direction. Making large changes in $\mathbf{z}(t)$ at such a condition can lead to worse states. The reasonable strategy in such a condition is to make $I_2(t)$ slightly more negative until $\Delta I_{\mathbf{z}(t)} < 0$ is achieved. This is done

by adjusting $\mathbf{z}(t)$ using $c_L(t) = c_2$, where c_2 is a small positive constant less than c_1.

The common Lyapunov-based update methods and the proposed update method as explained above have the same objective, i.e. to minimize the performance measure. However, they are very different in terms of the definition of the performance measure and the update timing.

In the common Lyapunov methods, the update rule is desired to minimize the Lyapunov function candidate, or equivalently, it is desired to achieve a negative derivative of the Lyapunov function candidate. Hence, the role of the Lyapunov function candidate can be considered similar to that of the performance measure. Unfortunately, the Lyapunov function candidate is not built to evaluate the adaptive control performance and therefore its real-time computation is unnecessary. It is commonly built using an assumed plant model, or at least an assumed model structure, and merely used to derive the update rule equation. Consequently, it is not computable and cannot provide real-time feedback information about whether the controller parameter vector is pointing in the right direction or not.

Although the common Lyapunov-based update rules can theoretically guarantee stable adaptive control, the performance of the adaptive control can suffer since it depends on the validity of the assumed model. When the assumed model deviates from the true one, the adjustment process of the controller parameters can lead to an undesired condition when the Lyapunov function increases, similar to condition 4, 5 or 6. In such a condition, the common Lyapunov-based update methods do not have any mechanism to deal with proper update timing. They keep on adjusting the controller parameter vector no matter if it is pointing in the wrong direction or not.

In the proposed update method, the control performance is measured using an evaluation function. Unlike a common Lyapunov function candidate, the proposed evaluation function is computable and can give the controller evaluative feedback information about whether the controller parameter vector is pointing in the right direction or not. Having real-time evaluative feedback information, the proposed update rule can update the direction of the controller parameter vector at the right timing by choosing the best $c_L(t)$ according to the condition occurring.

13 Simulations Results

In this section, the proposed adaptive fuzzy control is applied to solve the balancing problem of the benchmark inverted pendulum system, which has the following dynamics in Eq. (32):

$$\begin{aligned} \dot{s}_1(t) &= s_2(t) \\ \dot{s}_2(t) &= \frac{g \sin s_1(t) + \cos s_1(t)\left(\frac{-u(t) - mls_2^2(t)\sin s_1(t)}{m_C + m}\right)}{l\left(\frac{4}{3} - \frac{m(\cos s_1(t))^2}{m_C + m}\right)} \end{aligned} \tag{32}$$

where $s_1 = \theta$ is the angle of the pendulum to be balanced in the upright position (i.e. at zero degrees), $s_2 = \dot{\theta}$ is the angular velocity of the pendulum, u is the control force, g is the acceleration due to gravity, m_c is the mass of the cart, m is the mass of the pendulum, and l is the half length of the pendulum.

In the simulation, the following parameters were used

$$g = 9.81\ ms^2, m_c = 1\ kg, m = 0.1\ kg, l = 0.5\ m.$$

The inverted pendulum dynamics are numerically simulated using the 4th order Runge-Kutta integration method with a step size of $10\ ms$.

The fuzzy controller is desired to generate appropriate control forces capable of balancing the pendulum in the upright position. It is given two inputs: s_1 and s_2, and then generates u. s_1 and s_2 are defined to be within the range $[-30, 30]$ deg and $[-60, 60]$ deg/s, respectively. We define 5 Gaussian membership functions for the first input with the centers at $[-30, -20, 0, 20, 30]$ deg and the standard deviations of [20,10,10,10,20] deg, and for the second input with the centers at $[-60, -30,0,30,60]$ deg/s and the standard deviations of [30,15,10,15,30] deg/s. Any element of the state exceeding its range will be assigned with the maximum degree of membership (i.e. 1).

In the simulation, the following settings were chosen. The initial states were $s_1(0) = 15\ deg$ and $s_2(0) = 0\ rad/s$. The initial parameters of the fuzzy controller were $\mathbf{z}(0) = [0,0,\ldots,0]^T$. The design parameters were $\eta = 6000$, $\alpha = 10^{-5}$, $c_1 = 5$, and $c_2 = 0.2$.

Given the above membership functions for each input, there are 25 possible fuzzy rules, each of which has its own output parameter. Initialized with zero values, all those output parameters are the only fuzzy controller parameters to be adjusted simultaneously to enable the fuzzy controller to generate the appropriate control forces. There will be a heavy burden in making them appropriate, as any prior knowledge to initialize them is unavailable. Reducing

the number of fuzzy rules is possible and the balancing problem can be still solved. The author, however, is not primarily interested in solving the balancing problem of the inverted pendulum system using the proposed adaptive fuzzy controller. Rather, all the fuzzy controller parameters are initialized to a value of zero to make the control problem more difficult. Although many other well-developed adaptive control methods can be and have been successfully applied to the control problem addressed in this paper, they may not be successful with the simple settings used in this paper.

In the simulation, two control problems are considered, i.e. the balancing problem, where the goal state is fixed at $\mathbf{s}_{goal} = [0\ 0]^T$, and the tracking problem, where the goal state is changing, i.e. $\mathbf{s}_{goal} = [\frac{\pi}{30}\sin(t)\ 0]^T$. For comparison, the application of AFC to both control problems were simulated using the following three cases:

1. AFC with AGDM in Eq. (20), denoted by AFC
2. AFC with the update rule in Eq. (21), denoted by AFC/FD
3. AFC with the proposed update rule in Eq. (23), denoted by AFC/LM.

Note that cases 1 and 2 are the same as considered in [18], in which Eq. (21) was originally proposed, the update rule with failure detector. They were simulated again to be compared with the new update rule proposed in this paper, i.e. Eq. (23).

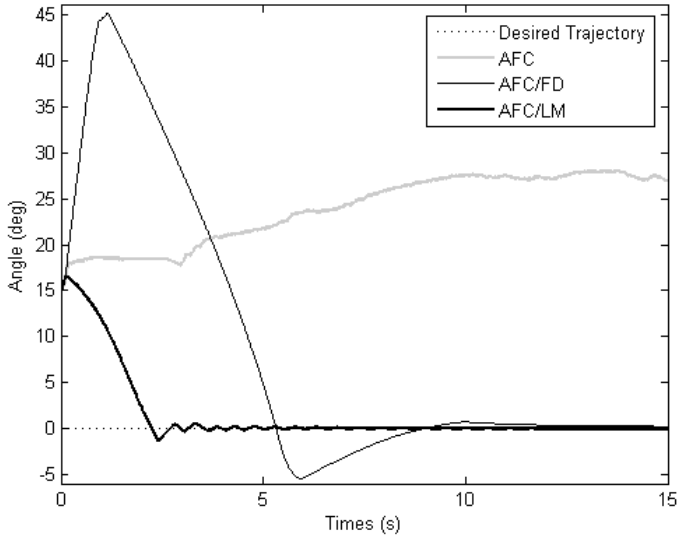

Figure 5 Balancing problem of inverted pendulum.

The simulation results are shown in Figures 5 and 6 for the balancing problem and in Figures 7 and 8 for the tracking problem. Figures 5 and 7 show the angle, while Figures 6 and 8 show the generated control actions. As can be seen in Figures 5 and 7, the pendulum converged to the desired goal state, but control with AFC always failed.

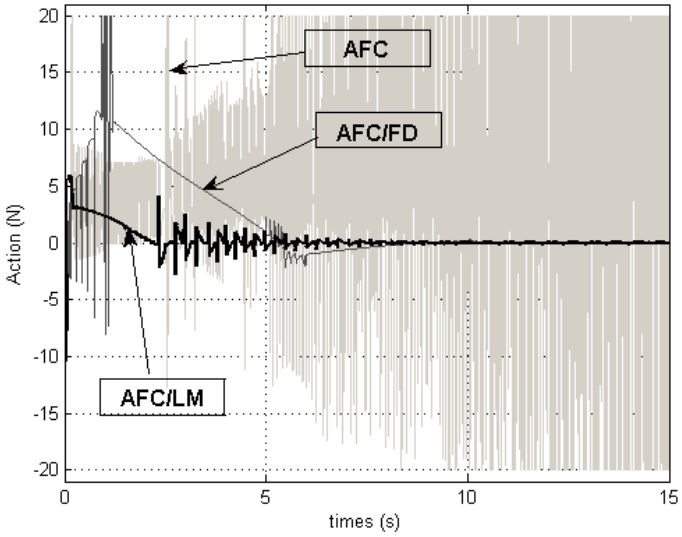

Figure 6 Control actions in balancing problem of inverted pendulum.

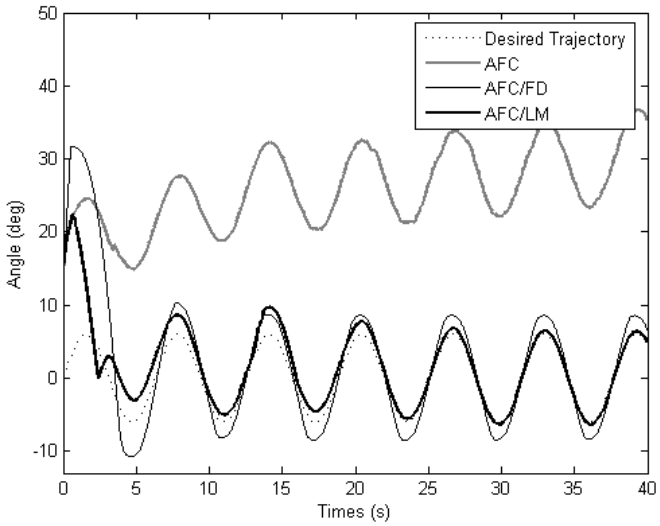

Figure 7 Tracking problems of inverted pendulum.

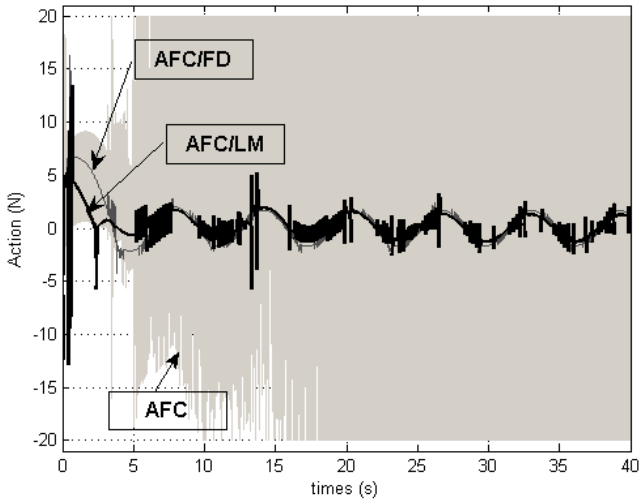

Figure 8 Control actions in tracking problem of inverted pendulum.

The simulation results in Figures 5 and 6 demonstrate that the update rule with the Lyapunov approach (see AFC/LM's performances) worked much better than that using the failure detector (see the AFC/FD's performances). In addition, the control actions in the case of AFC/LM were most effective, as shown in Figures 6 and 8. They took the least effort in achieving the goal state, particularly in the transient phase.

14 Conclusions

A model-free adaptive fuzzy controller that can improve its performance based on an evaluation function has been presented in this paper. The evaluation function, represented in terms of the distance between the current and the goal state and the weighted action, can tell the fuzzy controller the appropriate actions to execute in order to solve the control problem without using a plant model. The simulation results showed that the proposed adaptive fuzzy controller was effective in solving the problems of balancing and tracking an inverted pendulum system. The adaptive fuzzy control worked much better when its update rule was combined with the Lyapunov approach than with the failure detector only.

References

[1] Wang, L-X., *A Course In Fuzzy Systems And Control,* New Jersey, United States, Prentice-Hall International, Inc., 1997.

[2] Labiod, S. & Guerra, T.M., *Adaptive Fuzzy Control for a Class of SISO nonaffine Nonlinear Systems*, Fuzzy Sets and Systems, **158**(10), pp. 1098-1126, 2007a.
[3] Labiod, S. & Guerra, T.M., *Direct Adaptive Fuzzy Control for a Class of MIMO Nonlinear Systems*, International Journal of System Science, **38**(8), pp. 665-675.
[4] Labiod, S. & Guerra, T.M., *Indirect Adaptive Fuzzy Control for a Class of Nonaffine Nonlinear Systems with Unknown Control Directions*, International Journal of Control, Automation, and Systems, **8**(4), pp. 903-907, 2010.
[5] Bhasin, S., *Reinforcement Learning and Optimal Control Methods for Uncertain Nonlinear Systems*, Ph.D. Dissertation, University of Florida, United States, 2011.
[6] Sutton, R.S. & Barto, A.G., *Reinforcement Learning: An Introduction*, Cambridge, MA, United States, MIT Press, 1998.
[7] Åström, K.J. & Witternmark, B., *Adaptive Control*. Addison-Wesley Publishing Company, 1989.
[8] Lin, C-K., *A Reinforcement Learning Adaptive Fuzzy Controller for Robots,* Fuzzy sets and system, **137**(3), pp. 339-352, 2003.
[9] Nazaruddin, Y.Y., Naba, A. & Liong, T.H., *Modified Adaptive Fuzzy Control System Using Universal Supervisory Controller*, in Proc. of SCI/ISAS 2000, Orlando, USA, vol. IX, 2000.
[10] Oh, S.K., Pedrycz, W., Rho, S.B. & Ahn, T.C., *Parameters Estimation of Fuzzy Controller and its Application to Inverted Pendulum*," Engineering Applications of Artificial Intelligence, **17**(1), pp. 37-60, 2004.
[11] Park, J., Park, G., Kim, S. & Moon, C., *Direct Adaptive Self-Structuring Fuzzy Controller for Nonaffine Nonlinear System*, Fuzzy Sets and Systems, **153**(3), pp. 429-445, 2005.
[12] Liuzzo, S., Marino, R. & Tomei, P., *Adaptive Learning Control of Linear Systems by Output Error Feedback*, Automatica, **43**(4), pp. 669-676, 2007.
[13] Wu, L-B. & Yang, G-H., *Adaptive Fuzzy Tracking Control for a Class of Uncertain Nonaffine Nonlinear Systems with Dead-Zone Inputs*," Fuzzy Sets and Systems, **290**(C), pp. 1-21, May 2016.
[14] Chen, Y., Wei, Y., Liang, S., & Wang, Y., *Indirect Model Reference Adaptive Control for a Class of Fractional Order Systems,* Commun. Nonlinear Sci. Numer. Simul., **39**, pp. 458-471, 2016.
[15] Naba, A. & Miyashita, K., *Tuning Fuzzy Controller Using Approximated Evaluation Function*, in Proc. of the 4th IEEE International Workshop WSTST05, Muroran, Japan, pp. 113-122, 2005.
[16] Naba, A. & Miyashita, K., *Gradient-based Tuning of Fuzzy Controller with Approximated Evaluation Function*, in Proc. of Eleventh

International Fuzzy Systems Association (IFSA) World Congress, Beijing, China, pp. 671-676, 2005.

[17] Naba, A. & Miyashita, K., *FCAPS: Fuzzy Controller with Approximated Policy Search Approach*, Journal of Adv. Comput. Intelligence and Intelligent Informatics, **1**(1), pp. 84-92, 2006.

[18] Naba, A., *Adaptive Control with Approximated Policy Search Approach,* ITB Journal of Engineering Science, **42**(1), pp. 17-38, 2010.

[19] Sutton, R.S., *Learning to Predict by the Methods of Temporal Differences*, Machine Learning, **3**(1), pp. 9-44, 1988.

[20] Berenji, H.R. & Khedkar, P., *Learning and Tuning Fuzzy Logic Controllers Through Reinforcement*, IEEE, **3**(5), pp. 724–740, 1992.

[21] Berenji, H.R. & Khedkar, P., *Using Fuzzy Logic for Performance Evaluation in Reinforcement Learning*, International Journal of Approximate Reasoning, **18**, pp. 131–144, 1998.

[22] Sutton, R.S., McAllester, D., Singh, S. & Mansour, Y., *Policy Gradient Methods for Reinforcement Learning with Function Approximation*, Advances in Neural Information Processing System, **12**, pp. 1057-1063, 2000.

[23] Santamaria, J.C., Sutton, R.R. & Ram, A., *Experiment with Reinforcement Learning in Problems with Continuous State and Action Spaces*, Adaptive Behavior, **6**(2), pp. 163-218, 1998.

[24] Slotine, J-J.E. & Li, W., *Applied Nonlinear Control,* Englewood Cliffs New Jersey, United States, Prentice Hall, 1991.

[25] Baird, L.C. & Moore, A.W., *Gradient Descent for General Reinforcement Learning*, Advances in Neural Information Processing Systems 11, 1999.